МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чернігівський національний технологічний університет

ЗАСТОСУВАННЯ UML ДЛЯ МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторного практикуму
з дисципліни
"Об'єктно-орієнтований аналіз і проектування"
для студентів напряму підготовки
123 - “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
інформаційних і комп'ютерних систем

Протокол № 12 від 27.06.2018

Чернігів ЧНТУ 2018

Застосування uml для моделювання та проектування інформаційних систем Методичні вказівки до лабораторного практикуму з дисципліни „Об’єктно-орієнтований аналіз та проектування” для студентів напряму підготовки 123 - “Комп’ютерна інженерія”./ Укл. А.М. Акименко, І.В. Богдан, А.С. Посадська — Чернігів: ЧНТУ, 2018. — 37 с. укр. мовою.

Укладачі: Акименко Андрій Миколайович, кандидат фізико-математичних наук, доцент
Богдан Ірина Валентинівна, кандидат технічних наук, доцент
Посадська Аліна Сергіївна, кандидат технічних наук, ст. викладач

Відповідальний за випуск: Зайцев Сергій Васильович, завідуючий кафедрою інформаційних та комп’ютерних систем, доктор технічних наук, доцент

Рецензент: Нестеренко Сергій Олександрович, кандидат технічних наук, доцент кафедри інформаційних та комп’ютерних систем Чернігівського національного технологічного університету

ЗМІСТ

ПЕРЕДМОВА

Проектування інформаційних систем завжди починається з визначення мети проекту. Основне завдання будь-якого успішного проекту полягає в тому, щоб на момент запуску системи і протягом всього часу її експлуатації можна було забезпечити:

- необхідну функціональність системи і ступінь адаптації до умов її функціонування, що постійно змінюються;
- необхідну пропускну здатність системи;
- необхідний час реакції системи на запит;
- безвідмовну роботу системи;
- простоту експлуатації і підтримки системи;
- необхідну безпеку.

Проектування інформаційних систем охоплює три основні області:

- проектування об'єктів даних, які будуть реалізовані в базі даних;
- проектування програм, екранних форм, звітів, які будуть забезпечувати виконання запитів до даних;
- облік конкретного середовища або технології, а саме: топології мережі, конфігурації апаратних засобів, використовуваної архітектури (файл-сервер або клієнт-сервер), паралельної обробки, розподіленої обробки даних і т.ін.

В реальних умовах проектування - це пошук способу, який задовольняє вимогам функціональності системи засобами наявних технологій з урахуванням заданих обмежень.

До будь-якого проекту пред'являється ряд абсолютних вимог, наприклад, максимальний час розробки проекту, максимальні грошові вкладення в проект і т.ін. Одна зі складностей проектування полягає в тому, що воно не є таким структурованим завданням, як аналіз вимог до проекту або реалізація того чи іншого проектного рішення.

1 ЗАГАЛЬНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ

1.1 Мета лабораторного практикуму

Лабораторний практикум виконується при вивченні курсу "Об'єктний аналіз і проектування" і має на меті формування у студентів навичок в трьох напрямках:

- застосування відповідних методологій для розробки інформаційних систем і програмного забезпечення;
- застосування мови UML для моделювання та проектування інформаційних систем;
- застосування відповідного програмного інструментарію.

В "Загальні вказівки" винесені загальні для виконання всіх лабораторних робіт вимоги і правила.

1.2 Порядок виконання лабораторних робіт

Варіанти індивідуального завдання визначаються викладачем відповідно до списку індивідуальних завдань, розташованому в розділі 9 цих Методичних вказівок.

Для виконання всіх лабораторних робіт пропонується єдиний порядок, який передбачає наступні кроки:

1. Ознайомитися з постановкою завдання і вихідними даними.
2. Розробити запропоновану в роботі діаграму.
3. Реалізувати розроблену діаграму.
4. Зберегти файл моделі.
5. Скласти звіт про виконану роботу.

1.3 Зміст звіту про виконання лабораторних робіт

Звіт оформляється по кожній лабораторній роботі і складається з наступних розділів:

1. Тема лабораторної роботи.
2. Мета роботи.
3. Індивідуальне завдання.
4. Опис необхідних абстракцій (елементів діаграм).
5. Розроблена діаграма

2 ЛАБОРАТОРНА РОБОТА №1. ОПИС ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Мета роботи

Згідно варіанту, виконати опис предметної області проектованої програмної системи. Провести об'єктний аналіз отриманого опису і побудувати модель середовища за допомогою діаграми потоків даних (аналіз поведінки системи) і діаграми «сутність-зв'язок» (аналіз даних). Визначити призначення проектованої ІКС.

2.2 Теоретичні відомості

2.2.1 Аналіз предметної області

Етап аналізу передбачає докладне дослідження бізнес-процесів (функцій, визначених на етапі вибору стратегії) і інформації, необхідної для їх виконання (сутностей, їх атрибутів і зв'язків (відношень)). На цьому етапі створюється інформаційна модель системи.

Вся інформація про систему формалізується і уточнюється на етапі аналізу. Особливу увагу слід приділити повноті переданої інформації, аналізу інформації на предмет відсутності протиріч, а також пошуку інформації, яка не використовується взагалі або дублюється.

Аналітики збирають і фіксують інформацію в двох взаємопов'язаних формах:

- функції - інформація про події та процеси, які відбуваються в бізнесі;
- сутності - інформація про речі, які мають значення для організації і про яких щось відомо.

Двома класичними результатами аналізу є:

- ієрархія функцій, яка розбиває процес обробки на складові частини (що робиться і з чого це складається);
- модель «сутність-зв'язок» (Entry Relationship model, ER-модель), яка описує сутності, їх атрибути і зв'язки (відношення) між ними.

Ці результати є необхідними, але не достатніми. До достатніх результатів слід віднести діаграми потоків даних.

Нижче ми розглянемо методології структурного аналізу, що застосовуються частіше за все:

- діаграми «сутність-зв'язок» (Entity-Relationship Diagrams, ERD), які служать для формалізації інформації про сутності і їхні відношення;

– діаграми потоків даних (Data Flow Diagrams, DFD), які служать для формалізації представлення функцій системи.

2.2.2 Діаграми «сутність-зв'язок»

Діаграми "сутність-зв'язок" (ERD) призначені для розробки моделей даних і забезпечують стандартний спосіб визначення даних і відношень між ними. Фактично за допомогою ERD здійснюється деталізація сховищ даних проектованої системи, а також документуються сутності системи і способи їх взаємодії, включаючи ідентифікацію об'єктів, важливих для предметної області (сутностей), властивостей цих об'єктів (атрибутів) і їх відношень з іншими об'єктами (зв'язків).

Під сутністю (entity) розуміється довільна множина реальних або абстрактних об'єктів, кожен з яких володіє однаковими властивостями і характеристиками. У цьому випадку кожен даний об'єкт може бути екземпляром однієї і тільки однієї сутності, повинен мати унікальне ім'я або ідентифікатор, а також відрізнятися від інших екземплярів даної сутності. Прикладами сутностей можуть бути: банк, клієнт банку, комп'ютер, термінал. Кожна з сутностей може розглядатися з різним ступенем деталізації і на різному рівні абстракції, що визначається конкретною постановкою завдання. Для графічного представлення сутностей використовуються спеціальні позначення (рисунок 2.1).

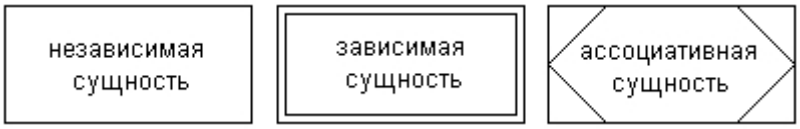

Рисунок 2.1 - Графічні зображення для позначення сутностей

Зв'язок (relationship) визначається як відношення або деяка асоціація між окремими сутностями. Прикладами зв'язків можуть бути родинні стосунки типу "батько-син" або виробничі відносини типу "начальник-підлеглий". Інший тип зв'язків задається відносинами "мати у власності" або "мати властивість". Різні типи зв'язків графічно зображаються у формі ромба з відповідним ім'ям даного зв'язку (рисунок 2.2).

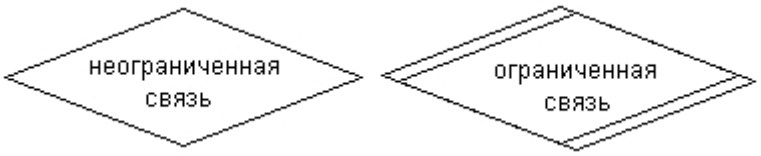

Рисунок 2.2 - Графічні зображення для позначення зв'язків

2.2.3 Діаграми потоків даних

Діаграми потоків даних являють собою інформаційну модель (DFD), основними компонентами якої є різні потоки даних, які переносять інформацію від однієї підсистеми до іншої. Кожна з підсистем виконує певні перет-

ворення вхідного потоку даних і передає результати обробки інформації у вигляді потоків даних для інших підсистем.

Основними компонентами діаграм потоків даних є:

- зовнішні сутності,
- накопичувачі даних або сховища,
- процеси,
- потоки даних,
- системи / підсистеми.

Зовнішня сутність є матеріальним об'єктом або фізичною особою, яка може виступати в якості джерела або приймача інформації. Прикладами зовнішніх сутностей можуть служити: клієнти організації, замовники, персонал, постачальники. Зовнішня сутність позначається прямокутником з тінню (рисунок 2.3), всередині якого вказується її ім'я. При цьому в якості імені рекомендується використовувати іменник у називному відмінку.

Рисунок 2.3 - Зображення зовнішньої сутності на діаграмі потоків даних

Процес являє собою сукупність операцій з перетворення вхідних потоків даних у вихідні відповідно до певного алгоритму або правила. Хоча фізично процес може бути реалізований різними способами, найбільш часто мається на увазі програмна реалізація процесу. Процес на діаграмі потоків даних зображується прямокутником із закругленими вершинами (рисунок 2.4), розділеним на три секції або поля горизонтальними лініями. Поле номера процесу служить для ідентифікації останнього. У середньому полі вказується ім'я процесу. Як ім'я рекомендовано використовувати дієслово в невизначеній формі з необхідними доповненнями. Нижнє поле містить вказівку на спосіб фізичної реалізації процесу.

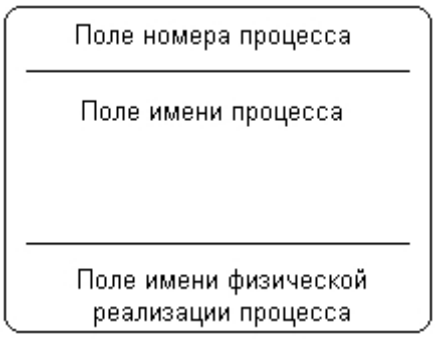

Рисунок 2.4 - Зображення процесу на діаграмі потоків даних

Накопичувач даних або сховище являє собою абстрактний пристрій або спосіб зберігання інформації, що перетікає між процесами. Передбачається, що дані можна в будь-який момент помістити в накопичувач і через деякий час витягнути, причому фізичні способи приміщення і вилучення даних можуть бути

довільними. Накопичувач даних на діаграмі потоків даних зображується прямокутником з двома полями (рисунок 2.5).

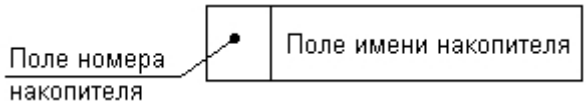

Рисунок 2.5 - Зображення накопичувача на діаграмі потоків даних

Потік даних визначає якісний характер інформації, переданої через деяке з'єднання від джерела до приймача. Потік даних на діаграмі DFD зображається лінією зі стрілкою на одному з її кінців, при цьому стрілка показує напрямок потоку даних. Кожен потік даних має своє власне ім'я, що відбиває його зміст.

Таким чином, інформаційна модель системи в нотації DFD будується у вигляді діаграм потоків даних, які графічно подаються з використанням відповідної системи позначень.

2.3 Зміст звіту

1. Найменування і мета роботи, номер варіанту.
2. Опис предметної області.
3. Розроблені діаграми потоків даних і «сутність-зв'язок».
4. Призначення програмної системи і опис її основних функцій.
5. Висновки.

2.4 Контрольні питання

1. Мета проведення об'єктного аналізу.
2. Призначення діаграми «сутність-зв'язок».
3. Основні елементи діаграми «сутність-зв'язок».
4. Призначення діаграми потоків даних.
5. Основні елементи діаграми потоків даних.

3 ЛАБОРАТОРНА РОБОТА №2. ОФОРМЛЕННЯ РЕЗУЛЬТАТІВ АНАЛІЗУ ЗА ДОПОМОГОЮ ДІАГРАМ UML

3.1 Мета роботи

Вивчити правила оформлення діаграми варіантів використання і діаграми аналізу. Навчиться виділяти особливості функціонального поведінки проектованої системи.

3.2 Теоретичні відомості

3.2.1 Побудова діаграми варіантів використання

Візуальне моделювання в UML можна представити як певний процес порівневого спуску від найбільш загальної і абстрактної моделі вихідної системи до логічної, а потім і до фізичної моделі відповідної програмної системи. Для досягнення цих цілей спочатку будується модель у формі діаграми варіантів використання (use case diagram), яка описує функціональне призначення системи або, іншими словами, те, що система буде робити в процесі свого функціонування.

Розробка діаграми варіантів використання переслідує мету:

- визначити спільні кордони і контекст модельованої предметної області на початкових етапах проектування системи;
- сформулювати загальні вимоги до функціонального поведінки проектованої системи;
- розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей;
- підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою так званих варіантів використання. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Це може бути людина, технічний пристрій або програма, які можуть служити джерелом впливу на систему, що моделюється, так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору.

Ціль варіанту використання полягає в тому, щоб визначити закінчений аспект або фрагмент поведінки деякої сутності без розкриття її внутрішньої структури. В якості такої сутності може виступати вихідна система або будь-який інший елемент моделі, який володіє власною поведінкою.

Кожен варіант використання відповідає окремому сервісу, який надає сутність або систему, що моделюється, по запиту користувача (актора), тобто визначає спосіб застосування цієї сутності. Сервіс, який ініціалізується за запи-

том користувача, є закінченою послідовністю дій. Це означає, що після того як система закінчить обробку запиту користувача, вона повинна повернутися в початковий стан, в якому готова до виконання наступних запитів.

Між компонентами діаграми варіантів використання можуть існувати різні відношення, які описують взаємодію екземплярів одних акторів і варіантів використання з екземплярами інших акторів і варіантів. Один актор може взаємодіяти з декількома варіантами використання. В цьому випадку цей актор звертається до кількох сервісів даної системи. У свою чергу один варіант використання може взаємодіяти з декількома акторами, надаючи для всіх них свій сервіс. Слід зауважити, що два варіанти використання, визначені для однієї і тієї ж сутності, не можуть взаємодіяти один з одним, оскільки кожен з них самостійно описує закінчений варіант використання цієї сутності.

В UML є кілька стандартних видів відношень між акторами і варіантами використання:

- відношення асоціації (association relationship),
- відношення розширення (extend relationship),
- відношення узагальнення (generalization relationship),
- відношення включення (include relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме за допомогою відносин розширення, узагальнення і включення.

Відношення асоціації

Стосовно діаграм варіантів використання асоціація специфікує семантичні особливості взаємодії акторів і варіантів використання в графічній моделі системи, тобто, це відношення встановлює, яку конкретну роль грає актор при взаємодії з екземпляром варіанту використання. На діаграмі варіантів використання відношення асоціації позначається суцільною лінією між актором і варіантом використання. Ця лінія може мати умовні позначення, такі як ім'я та кратність.

Кратність (multiplicity) асоціації вказується поряд з позначенням компонента діаграми, який є учасником даної асоціації, і характеризує кількість примірників даного компонента, які можуть виступати в якості елементів даної асоціації. Стосовно діаграм варіантів використання кратність має спеціальне позначення у формі однієї або декількох цифр і символу зірочка.

Для діаграм варіантів використання найпоширенішими є чотири основні форми запису кратності відношення асоціації:

- ціле невід'ємне число (включаючи 0). Призначено для вказівки кратності, яка є строго фіксованою для елемента відповідної асоціації. У цьому випадку кількість примірників акторів або варіантів використання, які можуть виступати в якості елементів відносини асоціації, в точності рівно зазначеному числу;
- два цілих невід'ємних числа, розділені двома крапками. Даний запис відповідає нотації для множини або інтервалу цілих чисел, яка застосовується в деяких мовах програмування для позначення меж ма-

сиву елементів. Цей запис слід розуміти як множину цілих невід'ємних чисел, які прямують у послідовно зростаючому порядку;

- два символи, розділені двома крапками. При цьому перший з них є цілим невід'ємним числом або 0, а другий - спеціальним символом «*», який позначає довільне кінцеве ціле невід'ємне число, значення якого невідоме на момент задання відповідного відношення асоціації;
- єдиний символ «*», який є скороченням запису інтервалу «0 .. *».

Якщо кратність відношення асоціації не вказана, то, за замовчуванням, приймається значення, рівне 1.

Відношення розширення

Відношення розширення визначає взаємозв'язок екземплярів окремого випадку використання з більш загальним варіантом, властивості якого визначаються на основі способу спільного об'єднання даних екземплярів. Відношення розширення є направленим і вказує, що стосовно окремих прикладів певного варіанту використання повинні бути виконані конкретні умови, визначені для розширення даного варіанту використання.

Відношення розширення між варіантами використання позначається пунктирною лінією зі стрілкою, спрямованою від того варіанту використання, який є розширенням для початкового варіанту використання.

Відношення розширення відзначає той факт, що один з варіантів використання може приєднувати до своєї поведінки деяку додаткову поведінку, визначену для іншого варіанту використання. Дане відношення включає в себе деяку умову і посилання на точки розширення в базовому варіанті використання. Щоб розширення мало місце, маэ бути виконана певна умова даного відношення.

Семантика відношення розширення визначається наступним чином. Якщо екземпляр варіанту використання виконує деяку послідовність дій, яка визначає його поведінку, і при цьому є точка розширення на екземпляр іншого варіанту використання, яка є першою з усіх точок розширення у початкового варіанту, то перевіряється умова даного відношення. Якщо умова виконується, вихідна послідовність дій розширюється за допомогою включення дій екземпляру іншого варіанту використання.

Відношення узагальнення

Відношення узагальнення служить для вказання того факту, що деякий варіант використання А може бути узагальнено до варіанту використання В. У цьому випадку варіант А буде спеціалізацією варіанту В. При цьому В називається предком або батьком по відношенню А, а варіант А - нащадком по відношенню до варіанта використання В. Слід підкреслити, що нащадок успадкує всі властивості і поведінку свого батька, а також може бути доповнений новими властивостями і особливостями поведінки. Графічно дане відношення позначається суцільною лінією зі стрілкою у формі незафарбованого трикутника, яка вказує на батьківський варіант використання.

Між окремими акторами також може існувати відношення узагальнення. Дане відношення є направленим і вказує на факт спеціалізації одних акторів щодо інших.

Відношення включення

Відношення включення між двома варіантами використання вказує, що деяка задана поведінка для одного варіанта використання включається як складовий компонент у послідовність поведінки іншого варіанту використання. Дане відношення є спрямованим бінарним відношенням у тому сенсі, що пара екземплярів варіантів використання завжди впорядкована у відношенні включення.

Семантика цього відношення визначається наступним чином. Коли екземпляр першого варіанту використання в процесі свого виконання досягає точки включення в послідовність поведінки екземпляра другого варіанту використання, екземпляр першого варіанту використання виконує послідовність дій, що визначає поведінку екземпляра другого варіанту використання, після чого продовжує виконання дій своєї поведінки. При цьому передбачається, що навіть якщо екземпляр першого варіанту використання може мати кілька екземплярів інших варіантів, що включаються в себе, виконувані ними дії повинні закінчитися до деякого моменту, після чого має бути продовжено виконання перерваних дій екземпляру першого варіанту використання відповідно до заданої для нього поведінки.

3.2.2 Побудова діаграми аналізу

Діаграма аналізу призначена для опису бізнес-процесів, що відбуваються в системі, моделі поведінки системи та її елементів. Дана діаграма є оптимальним засобом для опису бізнес-процесів та їх особливостей.

Діаграма аналізу являє собою спрощену версію діаграми процесів Еріксона-Пенкера, призначеної для моделювання бізнес-процесів в складних корпоративних системах.

Основними компонентами діаграми аналізу є:

- бізнес-процес,
- ресурс і інформація,
- подія,
- вихід,
- мета.

Бізнес-процес

Бізнес-процес являє собою набір дій, спрямованих на отримання конкретного результату для конкретного актора. Для кожного бізнес-процесу чітко заздалегідь визначаються входи і виходи. Позначення бізнес-процесу представлено на малюнку 3.1.

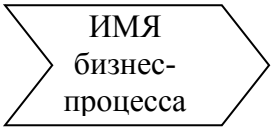

Рисунок 3.1 - Зображення бізнес-процесу на діаграмі аналізу

Ресурс та інформація

У процесі своєї реалізації бізнес-процес використовує ресурси. В якості ресурсів може виступати інформація від зовнішніх або внутрішніх джерел, від інших акторів або ж від інших бізнес-процесів.

На відміну від ресурсів, інформація не використовується бізнес-процесом, вона несе інформативний характер, пояснює чи уточнює певний нюанс.

Позначення ресурсу та інформації представлено на рисунку 3.2.

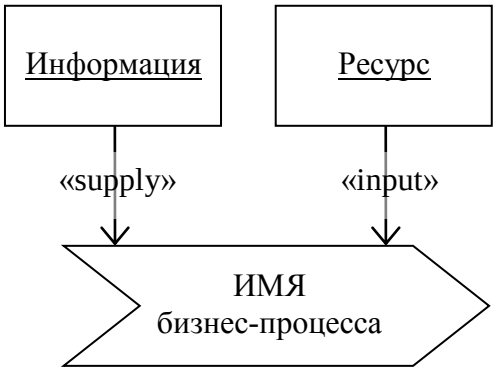

Рисунок 3.2 - Зображення ресурсу та інформації на діаграмі аналізу

Подія

Подією є який-небудь об'єкт, момент часу, дата, повідомлення або будь-який інший тригер, після спрацювання якого починається виконання бізнес-процесу. Позначення події представлено на малюнку 3.3.

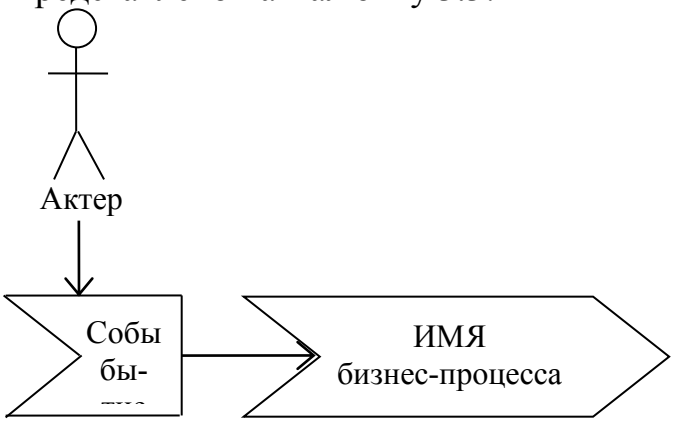

Рисунок 3.3 - Зображення ресурсу на діаграмі аналізу

Вихід

У результаті виконання бізнес-процесу, як правило, формується один або кілька вихідних результатів. Вихід може являти собою фізичний об'єкт (напри-

клад, звіт), перетворення наявних ресурсів в нових умовах (щоденний розклад або список) або загальний результат діяльності (наприклад, замовлення).

Позначення виходу представлено на рисунку 3.4.

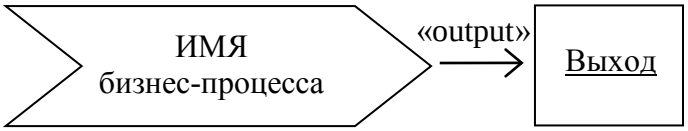

Рисунок 3.4 - Зображення виходу на діаграмі аналізу

Мета

Будь-який бізнес-процес має одну або кілька чітко визначених цілей. Саме мета є причиною організації і виконання бізнес-процесу.

Позначення цілі представлено на рисунку 3.5.

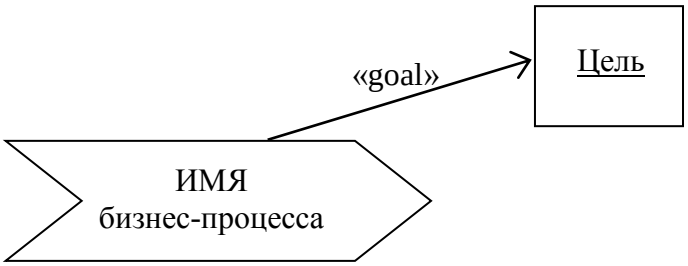

Рисунок 3.5 - Зображення мети на діаграмі аналізу

3.3 Зміст звіту

1. Найменування і мета роботи, номер варіанта.
2. Розроблені діаграми варіантів використання.
3. Специфікація поведінки елементів Use Case діаграми варіантів використання.
4. Специфікація інших елементів діаграми.
5. Розроблені діаграми аналізу.
6. Висновки.

3.4 Контрольні питання

1. Призначення діаграми варіантів використання.
2. Мета розробки діаграми варіантів використання.
3. Елементи діаграми варіантів використання. Актори.
4. Елементи діаграми варіантів використання. Відносини.
5. Елементи діаграми варіантів використання. Use Case.
6. Цілі розробки аналізу.
7. Елементи діаграми аналізу. Бізнес процеси.
8. Елементи діаграми аналізу. Ресурси та інформація.
9. Елементи діаграми аналізу. Події.
10. Елементи діаграми аналізу. Виходи.
11. Елементи діаграми аналізу. Цілі.

4 ЛАБОРАТОРНА РОБОТА №3. ДІАГРАМА КЛАСІВ

4.1 Мета роботи

Вивчити правила оформлення діаграми класів. Навчиться розробляти статичну структуру моделі системи в термінології класів об'єктно-орієнтованого програмування.

4.2 Теоретичні відомості

4.2.1 Діаграма класів

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відбивати різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру і типи відношень.

Діаграма класів складається з безлічі елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відношення між ними та їх складовими компонентами. При цьому окремі компоненти цієї діаграми можуть утворювати пакети для представлення більш загальної моделі системи. Якщо діаграма класів є частиною деякого пакета, то її компоненти повинні відповідати елементам цього пакета, включаючи можливі посилання на елементи з інших пакетів.

У загальному випадку пакет структурної статичної моделі може бути представлений у вигляді однієї або декількох діаграм класів. Декомпозиція деякого уявлення на окремі діаграми виконується з метою зручності та графічної візуалізації структурних взаємозв'язків предметної області. При цьому компоненти діаграми відповідають елементам статичної семантичної моделі. Модель системи, у свою чергу, повинна бути узгоджена з внутрішньою структурою класів, яка описується на мові UML.

У загальному випадку пакет структурної статичної моделі може бути представлений у вигляді однієї або декількох діаграм класів. Декомпозиція деякого уявлення на окремі діаграми виконується з метою зручності та графічної візуалізації структурних взаємозв'язків предметної області. При цьому компоненти діаграми відповідають елементам статичної семантичної моделі. Модель системи, у свою чергу, повинна бути узгоджена з внутрішньою структурою класів, яка описується на мові UML.

Клас

Клас (class) на мові UML служить для позначення множини об'єктів, які мають однакову структуру, поведінку і відношення з об'єктами з інших класів. Графічно клас зображується у вигляді прямокутника, який додатково може бу-

ти розділений горизонтальними лініями на розділи або секції. У цих розділах можуть зазначатися ім'я класу, атрибути (змінні) і операції (методи).

Обов'язковим елементом позначення класу є його ім'я. На початкових етапах розробки діаграми окремі класи можуть позначатися простим прямокутником із зазначенням тільки імені відповідного класу. У міру опрацювання окремих компонентів діаграми, описи класів доповнюються атрибутами і операціями.

Передбачається, що остаточний варіант діаграми містить найбільш повний опис класів, які складаються з трьох розділів або секцій.

Відношення між класами

Крім внутрішнього устрою або структури класів на відповідній діаграмі вказуються різні відношення між класами. При цьому сукупність типів таких відношень фіксована на мові UML і зумовлена семантикою цих типів відношень. Базовими відношеннями або зв'язками на мові UML є:

- відношення залежності (dependency relationship)
- відношення асоціації (association relationship)
- відношення узагальнення (generalization relationship)
- відношення реалізації (realization relationship)

Кожне з цих відношень має власне графічне представлення на діаграмі, яке відображає взаємозв'язки між об'єктами відповідних класів.

Відношення залежності

Відношення залежності в загальному випадку вказує деяке семантичне відношення між двома елементами моделі або двома множинами таких елементів, яке не є відношенням асоціації, узагальнення або реалізації. Відношення залежності використовується в такій ситуації, коли деяка зміна одного елемента моделі може потребувати зміни іншого залежного від нього елемента моделі.

Відношення асоціації

Відношення асоціації відповідає наявності деякого відношення між класами.

Найбільш простий випадок даного відношення - бінарна асоціація. Вона пов'язує в точності два класи і, як виняток, може пов'язувати клас з самим собою.

Відношення узагальнення

Відношення узагальнення є відношенням між більш загальним елементом (батьком або предком) і більш приватним або спеціальним елементом (дочірнім або нащадком). Дане відношення може використовуватися для подання взаємозв'язків між пакетами, класами, варіантами використання та іншими елементами мови UML.

Стосовно діаграми класів дане відношення описує ієрархічну будову класів та успадкування їх властивостей і поведінки. При цьому передбачається, що клас-нащадок має всі властивості і поведінку класу-предка, а також має свої власні властивості і поведінку, які відсутні у класа-предка.

Відношення реалізації

Відношення реалізації має місце між двома елементами моделі в тому випадку, якщо один елемент (клієнт) реалізує поведінку, задану іншим (постачальником). Відношення реалізації підрозділяється на:

- відношення агрегації (aggregation relationship)
- відношення композиції (composition relationship)

Відношення агрегації

Відношення агрегації має місце між декількома класами в тому випадку, якщо один з класів є деякою сутністю, що включає в себе в якості складових частин інші сутності.

Дане відношення має фундаментальне значення для опису структури складних систем, оскільки застосовується для представлення системних взаємозв'язків типу "частина-ціле". Розкриваючи внутрішню структуру системи, відношення агрегації показує, з яких компонентів складається система і як вони пов'язані між собою

Відношення композиції

Відношення композиції служить для виділення спеціальної форми відношення "частина-ціле", при якій складові частини в деякому сенсі перебувають всередині цілого. Специфіка взаємозв'язку між ними полягає в тому, що частини не можуть виступати окремо від цілого.

4.2.2 Рекомендації щодо побудови діаграми класів

Процес розробки діаграми класів займає центральне місце в ООАП складних систем. Від уміння правильно вибрати класи і встановити між ними взаємозв'язки часто залежить не тільки успіх процесу проектування, а й продуктивність виконання програми.

Після розробки діаграми класів процес ООАП може бути продовжений в двох напрямках. З одного боку, якщо поведінка системи тривіальна, то можна приступити до розробки діаграм кооперації і компонентів. Однак для складних динамічних систем поведінка найважливіший аспект їх функціонування. Деталізація поведінки здійснюється послідовно при розробці діаграм станів, послідовності і діяльності.

4.3 Зміст звіту

1. Найменування і мета роботи, номер варіанта.
2. Розроблені діаграми класів.
3. Опис елементів діаграми класів (включаючи відношення).
4. Висновки.

4.4 Контрольні питання

1. Призначення діаграми класів.

2. Цілі розробки діаграми класів.
3. Елементи діаграми класів. Класи.
4. Елементи діаграми класів. Відношення.
5. Елементи діаграми класів. Об'єкти.

5 ЛАБОРАТОРНА РОБОТА №4. ДІАГРАМИ ВЗАЄМОДІЇ

5.1 Мета роботи

Вивчити правила оформлення діаграм послідовності та кооперації. Вивчити особливості взаємодії об'єктів проектованої системи.

5.2 Теоретичні відомості

Функціональність елементів діаграми варіантів використання відображається графічно на діаграмах взаємодії. Ці діаграми містять об'єкти і сполучення між об'єктами, які показують реалізацію поведінки.

5.2.1 Діаграма послідовності

На діаграмі послідовності зображуються виключно ті об'єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні асоціації з іншими об'єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об'єктів в часі. При цьому діаграма послідовності має начебто два виміри. Одне - зліва направо у вигляді вертикальних ліній, кожна з яких зображує лінію життя окремого об'єкта, який бере участь у взаємодії. Графічно кожен об'єкт зображується прямокутником і розташовується у верхній частині своєї лінії життя.

Крайнім зліва на діаграмі зображується об'єкт, який є ініціатором взаємодії. Праворуч зображується інший об'єкт, який безпосередньо взаємодіє з першим. Таким чином, всі об'єкти на діаграмі послідовності утворюють деякий порядок, який визначається ступенем активності цих об'єктів при взаємодії один з одним.

Другий вимір діаграми послідовності - вертикальна тимчасова вісь, спрямована зверху вниз. Початковому моменту часу відповідає сама верхня частина діаграми. При цьому взаємодія об'єктів реалізуються за допомогою повідомлень, які надсилаються одними об'єктами іншим. Повідомлення зображуються у вигляді горизонтальних стрілок з ім'ям повідомлення і також утворюють, порядок за часом свого виникнення.

Лінія життя об'єкту

Лінія життя об'єкту (object lifeline) зображується пунктирною вертикальною лінією, асоційованою з єдиним об'єктом на діаграмі послідовності. Лінія життя служить для позначення періоду часу, протягом якого об'єкт існує в системі і, отже, може потенційно брати участь у всіх її взаємодіях. Якщо об'єкт існує в системі постійно, то і його лінія життя повинна тривати по всій площині діаграми послідовності від самої верхньої її частини до найнижчої.

Окремі об'єкти, виконавши свою роль в системі, можуть бути знищені, щоб звільнити займані ними ресурси. Для таких об'єктів лінія життя обриваєть-

ся в момент його знищення. Для позначення моменту знищення об'єкта в мові UML використовується спеціальний символ у формі латинської букви "X".

Фокус управління

У процесі функціонування об'єктно-орієнтованих систем одні об'єкти можуть перебувати в активному стані, безпосередньо виконуючи певні дії або в стані пасивного очікування повідомлень від інших об'єктів. Щоб явно виділити подібну активність об'єктів, в мові UML застосовується спеціальне поняття, яке отримало назву фокуса управління (focus of control). Фокус управління зображується у формі витягнутого вузького прямокутника, верхня сторона якого позначає початок отримання фокусу управління об'єкта (початок активності), а її нижня сторона - закінчення фокусу управління (закінчення активності).

У окремих випадках ініціатором взаємодії в системі може бути актор або зовнішній користувач. В цьому випадку актор зображується на діаграмі послідовності найпершим об'єктом зліва зі своїм фокусом управління. Найчастіше актор і його фокус управління існуватимуть в системі постійно, відзначаючи характерну для користувача активність в ініціюванні взаємодій з системою. При цьому сам актор може мати власне ім'я або залишатися анонімним.

Повідомлення

Як було зазначено вище, мета взаємодії в контексті мови UML полягає в тому, щоб специфікувати комунікацію між безліччю взаємодіючих об'єктів. Кожна взаємодія описується сукупністю повідомлень, якими обмінюються між собою об'єкти що беруть участь у цій взаємодії. У цьому сенсі повідомлення (message) являє собою закінчений фрагмент інформації, який відправляється одним об'єктом іншому. При цьому прийом повідомлення ініціює виконання певних дій, спрямованих на вирішення окремого завдання тим об'єктом, якому це повідомлення надіслано.

Таким чином, повідомлення не тільки передають деяку інформацію, але і вимагають або припускають від приймаючого об'єкта виконання очікуваних дій. Повідомлення можуть ініціювати виконання операцій об'єктом відповідного класу, а параметри цих операцій передаються разом з повідомленням. На діаграмі послідовності всі повідомлення впорядковані за часом свого виникнення в системі.

У такому контексті кожне повідомлення має напрямок від об'єкта, який ініціює та відправляє повідомлення, до об'єкту, який його отримує.

Зазвичай повідомлення зображуються горизонтальними стрілками, що з'єднують лінії життя або фокуси управління двох об'єктів на діаграмі послідовності.

У мові UML кожне повідомлення асоціюється з деяким дією, яка має бути виконаною прийнятим його об'єктом. При цьому дія може мати деякі аргументи або параметри, в залежності від конкретних значень яких може бути отриманий різний результат. Відповідні параметри матиме і повідомлення, що викликало цю дію. Більш того, значення параметрів окремих повідомлень можуть містити

умовні вирази, утворюючи розгалуження або альтернативні шляхи основного потоку управління.

5.2.2 Діаграма кооперації

Діаграма кооперації призначена для специфікації структурних аспектів взаємодії. Головна особливість діаграми кооперації заключається в можливості графічно представити не тільки послідовність взаємодії, але і всі структурні відносини між об'єктами, які беруть участь в цій взаємодії.

Перш за все, на діаграмі кооперації зображуються об'єкти, що беруть участь у взаємодії, вони містять ім'я об'єкту, його клас і, можливо, значення атрибутів. Далі, як і на діаграмі класів, вказуються асоціації між об'єктами у вигляді різних сполучних ліній. Додатково можуть бути зображені динамічні зв'язки - потоки повідомлень.

Таким чином, за допомогою діаграми кооперації можна описати повний контекст взаємодій як своєрідний часовий "зріз" сукупності об'єктів, які взаємодіють між собою для виконання певного завдання або бізнес-мети програмної системи.

Кооперація

Поняття кооперації (collaboration) є одним з фундаментальних понять в мові UML. Воно служить для позначення безлічі взаємодіючих з певною метою об'єктів в загальному контексті модельованої системи. Мета самої кооперації полягає в тому, щоб уточняти особливості реалізації окремих найбільш значущих операцій в системі. Кооперація визначає структуру поведінки системи в термінах взаємодії учасників цієї кооперації.

Кооперація може бути представлена на двох рівнях:

- на рівні специфікації - показує ролі класифікаторів і ролі асоціацій в розглянутій взаємодії;
- на рівні прикладів - вказує екземпляри і зв'язки, що утворюють окремі ролі в кооперації.

Діаграма кооперації рівня специфікації показує ролі, які виконують елементи, що беруть участь у взаємодії. Елементами кооперації на цьому рівні є класи і асоціації, які означають окремі ролі класифікаторів і асоціації між учасниками кооперації.

Діаграма кооперації рівня прикладів представляється сукупністю об'єктів (екземпляри класів) і зв'язків (екземпляри асоціацій). При цьому зв'язки доповнюються стрілками повідомлень. На даному рівні показуються тільки об'єкти, що мають безпосереднє відношення до реалізації операції або класифікатора.

5.3 Зміст звіту

1. Найменування і мета роботи, номер варіанта.
2. Розроблені діаграми послідовності.
3. Специфікація діаграм послідовності.
4. Розроблені діаграми кооперації рівня прикладів.

5. Висновки

5.4 Контрольні питання

1. Призначення діаграми послідовності.
2. Особливості діаграми послідовності.
3. Елементи діаграми послідовності. Об'єкти.
4. Елементи діаграми послідовності. Повідомлення.
5. Діаграма кооперації рівня специфікації.
6. Діаграма кооперації рівня прикладів.

6 ЛАБОРАТОРНА РОБОТА №5. ДІАГРАМИ ПОВЕДІНКИ

6.1 Мета роботи

Вивчити правила оформлення діаграм станів і діяльності. Навчитися виділяти в поведінці елемента системи окремі стани. За допомогою діаграми діяльності навчитися відображати особливості алгоритмів, що реалізують основні функції системи.

6.2 Теоретичні відомості

6.2.1 Діаграма станів

Для моделювання поведінки на логічному рівні в мові UML можуть використовуватися відразу кілька канонічних діаграм: станів, діяльності, послідовності і кооперації, кожна з яких фіксує увагу на окремому аспекті функціонування системи. На відміну від інших діаграм діаграма станів описує процес зміни станів тільки окремого елемента моделі (від окремого екземпляра класу до всієї системи в цілому). При цьому зміна стану елемента системи може бути викликано зовнішніми впливами з боку інших підсистем або ззовні. Саме для опису реакції елемента моделі на подібні зовнішні впливи і використовуються діаграми станів.

Головне призначення цієї діаграми - описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма станів представляє динамічну поведінку сутностей, на основі специфікацій їх реакції на сприйняття деяких конкретних подій. Системи, які реагують на зовнішні впливи від інших систем або від користувачів, іноді називають реактивними. Якщо такі дії ініціюються в довільні випадкові моменти часу, то говорять про асинхронну поведінку моделі.

Стан

Поняття стану (state) є фундаментальним не тільки в мета-моделі мови UML, але і в прикладному системному аналізі.

У мові UML під станом розуміється абстрактний метаклас, який використовується для моделювання окремої ситуації, протягом якої має місце виконання деякої умови. Стан може бути задано у вигляді набору конкретних значень атрибутів класу або об'єкта, при цьому зміна їх окремих значень буде відображати зміну стану модельованого класу або об'єкта.

Перехід

Простий перехід (simple transition) являє собою відношення між двома послідовними станами, яке вказує на факт зміни одного стану іншим. Перебування модельованого об'єкта в першому стані може супроводжуватися виконанням деяких дій, а перехід в другий стан буде можливий після завершення

цих дій, а також після задоволення деяких додаткових умов. У цьому випадку говорять, що перехід спрацьовує. До спрацьовування переходу об'єкт знаходиться в попередньому від нього стані, званим вихідним станом, або в джерелі, а після його спрацьовування об'єкт знаходиться в подальшому від нього стані (цільовому стані).

Перехід здійснюється при настанні деякої події: закінчення виконання діяльності (do activity), отриманні об'єктом повідомлення або прийомом сигналу. На переході вказується назва події. Крім того, на переході можуть вказуватися дії, виконані об'єктом у відповідь на зовнішні події при переході з одного стану в інший. Спрацьовування переходу може залежати не тільки від настання деякої події, а й від виконання певної умови, яку називають сторожовою. Об'єкт перейде з одного стану в інший в тому випадку, якщо відбулася вказана подія і сторожова умова прийняла значення "істина".

6.2.2 Діаграма діяльності

При моделюванні поведінки проектованої або аналізованої системи виникає необхідність не тільки представити процес зміни її станів, але і деталізувати особливості алгоритмічної і логічної реалізації виконуваних системою операцій. Традиційно для цієї мети використовувалися блок-схеми або структурні схеми алгоритмів.

Для моделювання процесу виконання операцій в мові UML використовуються діаграми діяльності. Графічна нотація, яка в них застосовується багато в чому схожа на нотацію діаграми станів, оскільки на діаграмах діяльності також присутні позначення станів і переходів. Відмінність полягає в семантиці станів, які використовуються для представлення не діяльностей, а дій, і у відсутності на переходах сигнатури подій. Кожен стан на діаграмі діяльності відповідає виконанню деякої елементарної операції, а перехід в наступний стан спрацьовує тільки при завершенні цієї операції в попередньому стані. Фактично, діаграми діяльності можна вважати окремим випадком діаграм станів.

6.2.3 Рекомендації з побудови діаграм поведінки

Діаграма станів

За своїм призначенням діаграма станів не є обов'язковим представленням в моделі і начебто "приєднується" до того елемента, який має нетривіальну поведінку протягом свого життєвого циклу. Наявність у системи декількох нетривіальних станів, що відрізняються від «справний - несправний», служить ознакою необхідності побудови діаграми станів. В якості початкового варіанту діаграми станів, якщо немає очевидних міркувань з приводу станів об'єкта, можна скористатися цими суперстанами, розглядаючи їх як складові і деталізуючи їх внутрішню структуру в міру розгляду логіки поведінки об'єкта. При розробці діаграми станів потрібно постійно стежити, щоб об'єкт в кожен момент міг перебувати тільки в єдиному стані. Якщо це не так, то ця обставина може бути як

наслідком помилки, так і неявною ознакою наявності паралельності в поведінці модельованого об'єкта.

Діаграма діяльності

Діаграми діяльності відіграють важливу роль в розумінні процесів, що реалізують алгоритми виконання операцій класів і потоків управління в системі, що моделюється.

Зміст діаграми діяльності багато в чому нагадує діаграму станів, хоча і не є тотожним їй. Зокрема, ця діаграма будується для окремого класу, варіанту використання окремої операції класу або цілої підсистеми.

У разі типового проекту більшість деталей реалізації дій можуть бути відомі заздалегідь на основі аналізу існуючих систем або попереднього досвіду розробки систем-прототипів. Використання типових рішень може істотно скоротити час розробки і уникнути можливих помилок при реалізації проекту.

При розробці проекту нової системи, процес функціонування якої базується на нових технологічних рішеннях, ситуація складніша. А саме, до початку роботи над проектом можуть бути невідомі не тільки деталі реалізації окремих діяльностей, а й сам зміст цих діяльностей стає предметом розробки.

6.3 Зміст звіту

1. Найменування і мета роботи, номер варіанта.
2. Розроблені діаграми станів.
3. Специфікація діаграм станів.
4. Розроблені діаграми діяльності.
5. Висновки.

6.4 Контрольні питання

1. Призначення діаграми станів.
2. Особливості діаграми станів.
3. Елементи діаграми станів. Стани.
4. Елементи діаграми станів. Переходи.
5. Діаграма діяльності.

7 ЛАБОРАТОРНА РОБОТА №6. ДІАГРАМИ КОМПОНЕНТІВ І РОЗГОРТАННЯ

7.1 Мета роботи

Вивчити правила оформлення діаграми компонентів. Навчиться розробляти конкретну реалізацію проекту в формі програмного коду.

7.2 Теоретичні відомості

7.2.1 Представлення компонентів

Усі розглянуті раніше діаграми відносилися до логічного рівня представлення. Особливість логічного представлення полягає в тому, що різні елементи логічного представлення, такі як класи, асоціації, стани, повідомлення, не існують матеріально або фізично, вони лише відображають наше розуміння структури фізичної системи або аспекти її поведінки.

Повний проект програмної системи являє собою сукупність моделей логічного і фізичного представлень, які повинні бути узгоджені між собою. У мові UML для фізичного представлення моделей систем використовуються так звані діаграми реалізації (implementation diagrams), які включають в себе дві окремі канонічні діаграми: діаграму компонентів і діаграму розгортання.

Діаграма компонентів описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код.

Діаграма компонентів розробляється для наступних цілей:

1. Візуалізація загальної структури вихідного коду програмної системи.
2. Специфікації виконуваного варіанту програмної системи.
3. Забезпечення багаторазового використання окремих фрагментів програмного коду.
4. Представлення концептуальної і фізичної схем баз даних.

Діаграма компонентів забезпечує узгоджений перехід від логічного представлення до конкретної реалізації проекту у формі програмного коду. Одні компоненти можуть існувати тільки на етапі компіляції програмного коду, інші - на етапі його виконання. Діаграма компонентів відображає загальні залежності між компонентами, розглядаючи останні як класифікатори.

Компоненти

Для представлення фізичних сутностей в мові UML застосовується спеціальний термін - компонент (component). Компонент реалізує деякий набір інтерфейсів і служить для загального позначення елементів фізичного представлення моделі.

Компонент надає організацію в рамках фізичного пакету асоційованим з ним елементів моделі. Як класифікатор, компонент може мати також свої власні властивості, такі як атрибути і операції.

Залежності

Залежність не є асоціацією, а служить для представлення тільки факту наявності такого зв'язку, коли зміна одного елемента моделі впливає або призводить до зміни іншого елемента моделі. Відношення залежності на діаграмі компонентів зображується пунктирною лінією зі стрілкою, спрямованої від клієнта (залежного елементу) до джерела (незалежного елементу).

Залежності можуть відображати зв'язки модулів програми на етапі компіляції і генерації об'єктного коду. Стосовно діаграми компонентів залежності можуть пов'язувати компоненти і імпортовані цим компонентом інтерфейси, а також різні види компонентів між собою.

Також на діаграмі можуть бути представлені відношення залежності між компонентами і реалізованими в них класами. Ця інформація має важливе значення для забезпечення узгодження логічного і фізичного представлень моделі системи.

7.2.2 Рекомендації щодо побудови діаграми компонентів

Розробка діаграми компонентів припускає використання інформації, як про логічне представлення моделі системи, так і про особливості її фізичної реалізації. До початку розробки необхідно визначитися з вибором мовної платформи і операційної системи.

Після цього можна приступати до загальної структуризації діаграми компонентів. В першу чергу необхідно вирішити з яких фізичних частин (файлів) буде складатися програмна система. На цьому етапі слід звернути увагу на таку реалізацію системи, яка забезпечувала б не тільки можливість повторного використання коду за рахунок раціональної декомпозиції компонентів, але і створення об'єктів тільки при їх необхідності.

Після загальної структуризації фізичного представлення системи необхідно доповнити модель інтерфейсами і схемами бази даних. При розробці інтерфейсів слід звертати увагу на узгодження різних частин програмної системи. Включення в модель схеми бази даних передбачає специфікацію окремих таблиць і встановлення інформаційних зв'язків між таблицями.

7.2.3 Діаграма розгортання

Фізичне представлення програмної системи не може бути повним, якщо відсутня інформація про те, на якій платформі і на яких обчислювальних засобах вона реалізована. Для цього є кілька причин:

- складні програмні системи можуть реалізовуватися в мережевому варіанті на різних обчислювальних платформах і технологіях доступу до розподілених баз даних;

- інтеграція програмної системи з Інтернетом визначає необхідність вирішення додаткових питань при проектуванні системи, таких як забезпечення безпеки, стійкості доступу до інформації і т.д.
- технології доступу і обробки даних в схемі "клієнт-сервер" вимагає розміщення великих баз даних в різних сегментах корпоративної мережі, їх резервного копіювання для забезпечення необхідної продуктивності системи в цілому.

Діаграма розгортання (синонім - діаграма розміщення) застосовується для представлення загальної конфігурації і топології розподіленої програмної системи і містить розподіл компонентів за окремими вузлами системи. Крім того, діаграма розгортання показує наявність фізичних з'єднань - маршрутів передачі інформації між апаратними пристроями, задіяними в реалізації системи.

Діаграма розгортання призначена для візуалізації елементів і компонентів програми, які існують лише на етапі її виконання (runtime). При цьому представляються тільки компоненти-екземпляри програми, які є виконуваними файлами або динамічними бібліотеками.

Отже, перерахуємо цілі розробки діаграми розгортання:

- визначити розподіл компонентів системи за її фізичними вузлами;
- показати фізичні зв'язки між усіма вузлами реалізації системи на етапі її виконання;
- виявити вузькі місця системи і реконфігурувати її топологію для досягнення необхідної продуктивності.

Вузол

Вузол (node) являє собою деякий фізично існуючий елемент системи, що володіє деяким обчислювальним ресурсом. В якості обчислювального ресурсу вузла може розглядатися наявність хоча б деякого об'єму електронної пам'яті і / або процесора. В останній версії мови UML поняття вузла розширено і може включати в себе не тільки обчислювальні пристрої (процесори), а й інші механічні або електронні пристрої.

Дозволено показувати на діаграмі розгортання вузли з вкладеними зображеннями компонентів. Важливо пам'ятати, що в якості таких вкладених компонентів можуть виступати тільки виконувані компоненти.

З'єднання

Крім, власне, зображень вузлів на діаграмі розгортання вказуються відносини між ними. В якості відносин виступають фізичні з'єднання між вузлами і залежності між вузлами і компонентами, зображення яких теж можуть бути присутніми на діаграмах розгортання.

З'єднання є різновидом асоціації і зображуються відрізками ліній без стрілок. Наявність такої лінії вказує на необхідність організації фізичного каналу для обміну інформацією між відповідними вузлами. Характер з'єднання мо-

же бути додатково специфіковано приміткою, поміченою значенням або обмеженням.

Діаграми розгортання можуть мати більш складну структуру, яка включає вкладені компоненти, інтерфейси та інші апаратні пристрої.

7.2.4 Рекомендації щодо побудови діаграми розгортання

Розробка діаграми розгортання починається з ідентифікації всіх апаратних, механічних та інших типів пристроїв, які необхідні для виконання системою всіх своїх функцій. В першу чергу специфікуються обчислювальні вузли системи, що володіють пам'яттю і / або процесором.

Подальша побудова діаграми розгортання пов'язана з розміщенням всіх виконуваних компонентів діаграми за вузлами системи. Якщо окремі виконувані компоненти виявилися не розміщеними, то подібна ситуація повинна бути виключена шляхом введення в модель додаткових вузлів, що містять процесор і пам'ять.

При розробці простих програм, які виконуються локально на одному комп'ютері, так само як і в випадку діаграми компонентів, необхідність в діаграмі розгортання може взагалі бути відсутньою. У більш складних ситуаціях діаграма розгортання будується для таких додатків, як:

- моделювання програмних систем, що реалізують технологію доступу до даних "клієнт-сервер";
- моделювання неоднорідних розподілених архітектур (корпоративні мережі);
- моделювання системи з вбудованими мікропроцесорами, які можуть функціонувати автономно.

Як правило, розробка діаграми розгортання здійснюється на завершальному етапі ООАП, що характеризує закінчення фази проектування фізичного представлення. З іншого боку, діаграма розгортання може будуватися для аналізу існуючої системи з метою її подальшого аналізу і модифікації. При цьому аналіз передбачає розробку цієї діаграми на його початкових етапах, що характеризує загальний напрямок аналізу від фізичного представлення до логічного.

7.3 Зміст звіту

1. Найменування і мета роботи, номер варіанта.
2. Розроблена діаграма компонентів.
3. Специфікація діаграми компонентів.
4. Розроблена діаграма розгортання.
5. Специфікація діаграми розгортання.
6. Висновки.

7.4 Контрольні питання

1. Фізична модель програмної системи.

2. Призначення діаграми компонентів.
3. Цілі розробки діаграми компонентів.
4. Елементи діаграми компонентів. Компоненти.
5. Елементи діаграми компонентів. Залежності.
6. Призначення діаграми розгортання.
7. Цілі розробки діаграми розгортання.
8. Елементи діаграми розгортання. Вузол.
9. Елементи діаграми розгортання. З'єднання.

8 СПИСОК ІНДИВІДУАЛЬНИХ ВАРІАНТІВ ЗАВДАНЬ СТУДЕНТІВ

1. Розробити модель ІКС бібліотеки (актори - керівник, бібліотекар, читач)
2. Розробити модель ІКС рекламної фірми (актори - керівник, співробітник по роботі з клієнтами, клієнт)
3. Розробити модель ІКС відеосалону (актори - керівник, співробітник, клієнт)
4. Розробити модель ІКС магазину парфумерії (актори - керівник, співробітник, клієнт)
5. Розробити модель ІКС ресторану (актори - керівник, офіціант, клієнт)
6. Розробити модель ІКС організації по роботі з абітурієнтами (актори - керівник, співробітник організації, абітурієнт)
7. Розробити модель ІКС середньої школи (актори - директор, викладач, учень)
8. Розробити модель ІКС провайдера Інтернет (актори - керівник, співробітник по роботі з клієнтами, клієнт)
9. Розробити модель ІКС роботи військкомату (актори - керівник, співробітник по роботі з призовниками, призовник)
10. Розробити модель ІКС роботи центру зайнятості (актори - керівник, співробітник по роботі з безробітними, безробітний)
11. Розробити модель ІКС системи охорони підприємства (актори - керівник підприємства, співробітник підприємства, охоронець)
12. Розробити модель ІКС роботи університету (актори - ректор, декан, студент)
13. Розробити модель ІКС супермаркету (актори - директор, касир, покупець)
14. Розробити модель ІКС чемпіонату з хокею (актори - співробітник по роботі з хокеїстами, хокеїст, голова федерації хокею)
15. Розробити модель ІКС олімпійських ігор (актори - уболівальник, співробітник по роботі з уболівальниками, керівник олімпійського комітету)
16. Розробити модель ІКС зоопарку (актори - відвідувач, директор, касир)
17. Розробити модель ІКС театру (актори - актор, директор, адміністратор)
18. Розробити модель ІКС страхового агентства (актори - клієнт, директор, юрист)
19. Розробити модель ІКС весільного салону (актори - директор, співробітник по роботі з клієнтами, клієнт)
20. Розробити модель ІКС туристичної фірми (актори - директор, співробітник по роботі з клієнтами, клієнт)
21. Розробити модель ІКС перукарні (актори - керівник, перукар, клієнт)
22. Розробити модель ІКС піцерії (актори - керівник, офіціант, клієнт)
23. Розробити модель ІКС аукціонного дому (актори - керівник, співробітник, аукціоніст)
24. Розробити модель ІКС автопарк (актори - директор, клієнт, водій)
25. Розробити модель ІКС салону з продажу мобільних телефонів (актори - директор, продавець-консультант, клієнт)
26. Розробити модель ІКС кінологічного клубу (актори - керівник, кінолог, власник тварини)

27. Розробити модель ІКС дитячого садка (актори - директор, вихователь, батько)
28. Розробити модель ІКС управління програмними проектами (актори - керівник фірми, розробник, менеджер проектів)
29. Розробити модель ІКС командної розробки курсових проектів (актори - ректор, викладач, завідувач кафедри)
30. Розробити модель ІКС Міністерства освіти (актори - міністр освіти, співробітник міністерства, президент)
31. Розробити модель ІКС вокзалу (актори - начальник, машиніст, пасажир)
32. Розробити модель ІКС благодійного фонду (актори - керівник, меценат, співробітник по роботі з клієнтами)
33. Розробити модель ІКС обліку ДАІ (актори - адвокат, даїшник, свідок правопорушення)
34. Розробити модель ІКС роботи комунального підприємства (актори - директор, диспетчер, мешканець)
35. Розробити модель ІКС ремонтної майстерні (актори - директор, майстер, клієнт)
36. Розробити модель ІКС тролейбусного управління (актори - начальник, водій, технічний працівник)
37. Розробити модель ІКС житлово-комунального підприємства (актори - начальник, диспетчер, сантехнік)
38. Розробити модель ІКС кур'єрського агентства (актори - начальник, замовник кур'єрських послуг, співробітник по роботі з клієнтами)
39. Розробити модель ІКС агентства по догляду за людьми похилого віку (актори - керівник, людина похилого віку, психолог)
40. Розробити модель ІКС аптечного фонду (актори - начальник, клієнт, фармацефт)
41. Розробити модель ІКС надання послуг мобільного оператора (актори - начальник, споживач, технічний працівник)
42. Розробити ІКС кінотеатру (актори - директор, кіноман, касир)
43. Розробити модель ІКС планетарію (актори - керівник, екскурсовод, відвідувач)
44. Розробити модель ІКС парку розваг (актори - директор, відвідувач, касир)
45. Розробити модель ІКС магазину побутової техніки (актори - директор, клієнт, продавець)
46. Розробити модель ІКС банку (актори - голова, касир, клієнт)
47. Розробити модель ІКС кафе (актори - начальник, офіціант, відвідувач)
48. Розробити модель ІКС міських електричних мереж (актори - начальник, електрик, диспетчер)
49. Розробити модель ІКС РАЦСу (ЗАГСа) (актори - керівник, співробітник відділу прийому заяв, бажаючий одружитися)
50. Розробити модель ІКС фітнес-клубу (актори - керівник, тренер, клієнт)
51. Розробити модель ІКС роботи відділу кредитування банку (актори - голова, клієнт, співробітник, який оформляє кредити)

52. Розробити модель ІКС відеопрокату (актори - директор, співробітник, клієнт)
53. Розробити модель ІКС залу ігрових автоматів (актори - директор, технік-налаштовувач, гравець)
54. Розробити модель ІКС роботи відділу кадрів електро-механічного заводу (актори - начальник відділу кадрів, співробітник відділу кадрів, представник біржи праці)
55. Розробити модель ІКС фабрики з виробництва музичних інструментів (актори - керівник, співробітник, клієнт)
56. Розробити модель ІКС маркетингової фірми (актори - маркетолог, співробітник по роботі з клієнтами, клієнт)
57. Розробити модель ІКС лижної бази (актори - директор, тренер, технічний працівник)
58. Розробити модель ІКС авіакомпанії (актори - директор, пілот, пасажир)
59. Розробити модель ІКС агентства нерухомості (актори - керівник, ріелтор, юрист)
60. Розробити модель ІКС магазину спортивних товарів (актори - керівник, продавець-консультант, клієнт)
61. Розробити модель ІКС міграційної служби (актори - начальник, юрист, мігрант)
62. Розробити модель ІКС обліку пацієнтів поліклініки (актори - хворий, працівник реєстратури, лікар)
63. Розробити модель ІКС складу торгової фірми (актори - вантажник, комірник, водій транспорту з перевезення продукції з/на склад торгової фірми)
64. Розробити модель ІКС меблевого магазину (актори - директор, продавець-консультант, клієнт)
65. Розробити модель ІКС підприємства по створенню меблів (актори - керівник, клієнт, менеджер відділу продажів)
66. Розробити модель ІКС ательє з пошиття пальто (актори - директор, клієнт, швачка)
67. Розробити модель ІКС пральні (актори - клієнт, прачка, співробітник служби доставки)
68. Розробити модель ІКС хімчистки (актори - клієнт, співробітник, який працює з миючим апаратом, співробітник служби доставки)
69. Розробити модель ІКС автосервісу (актори - керівник, автомеханік, клієнт)
70. Розробити модель ІКС лізингу автомобілів (актори - керівник, клієнт, співробітник по роботі з клієнтами)
71. Розробити модель ІКС магазину товарів для туризму й відпочинку (актори - директор, продавець-консультант, клієнт)
72. Розробити модель ІКС деревообробного підприємства (актори - замовник, столяр, співробітник по роботі з клієнтами)
73. Розробити модель ІКС книгарні (актори - директор, продавець-консультант, клієнт)
74. Розробити модель ІКС букмекерської контори (актори - начальник, співробітник, який відповідає за прийом ставок, бажаючий зробити ставку)

75. Розробити модель ІКС травматологічного пункту (актори - медична сестра, хворий, лікар)
76. Розробити модель ІКС центру тестування випускників (актори - батько випускника, співробітник центру, випускник)
77. Розробити модель ІКС клініки пластичної хірургії (актори - завідувач, пластичний хірург, бажаючий зробити пластичну операцію)
78. Розробити модель ІКС центру допомоги ветеранам (актори - керівник, співробітник центру, ветеран)
79. Розробити модель ІКС центру допомоги багатодітним сім'ям (актори - керівник, співробітник центру, представник багатодітної сім'ї)
80. Розробити модель ІКС дитячого садка (актори - керівник, кухар, батько)
81. Розробити модель ІКС історичного музею (актори - керівник, співробітник, відвідувач)
82. Розробити модель ІКС роботи паспортного столу (актори - начальник, клієнт, співробітник по прийому заявок)
83. Розробити модель ІКС роботи приймальні міського голови (актори - керівник, працівник приймальні, громадянин)
84. Розробити модель ІКС роботи магазину комп'ютерної техніки (актори - директор, продавець-консультант, клієнт)
85. Розробити модель ІКС організації захисту тварин (актори - керівник, ветеринар, власник тварини)
86. Розробити модель ІКС роботи пенсійного фонду (актори - начальник, співробітник по розботі з клієнтами, пенсіонер)
87. Розробити модель ІКС агентства знайомств (актори - керівник, співробітник по роботі з клієнтами, клієнт)
88. Розробити модель ІКС роботи Льодової арени (актори - керівник, охоронець, відвідувач)
89. Розробити модель ІКС роботи парку розваг (актори - керівник, касир, відвідувач)
90. Розробити модель ІКС роботи річкового вокзалу (актори - пасажир, капітан, касир)

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. А. Леоненков. Самоучитель UML. Эффективный инструмент моделирования информационных систем. – BHV-Санкт-Петербург, 2001.- 304с.
2. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. 2-е изд.- М. : "Бином", 1999 г.- 560с.
3. Джеймс Рамбо, Айвар Якобсон, Грэди Буч. UML. Специальный справочник. – Питер, 2002. – 656с.
4. М.Фаулер, К.Скотт. UML в кратком изложении. Применение стандартного языка объектного моделирования. – М.: Мир, 1999. – 192с.
5. С.А.Трофимов. CASE-технологии. Практическая работа в Rational Rose. – Бином, 2002.-272с.
6. Терри Кватрани. Rational Rose 2000 и UML. Визуальное моделирование. – ДМК, 2001.- 176с.
7. Язык UML. Руководство пользователя. – ДМК, 2000.- 432с.